

REMARKS

The Office Action of April 1, 2005 has been carefully considered. In response thereto, the claims have been amended as set forth above.

Claims 1-5 were rejected as being anticipated by the O'Connor. Claims 1, 4 and 6 have been amended to more clearly distinguish over the cited reference. Reconsideration is respectfully requested.

The present invention relates, in one aspect, to a computational method in which one functional unit, during execution of an instruction, outputs data to or receives input data from another functional unit *during execution of the instruction*, where the functional units share a common memory, e.g., a micro code memory. Claims 1, 4 and 6 have been amended to make clear that such input/output occurs during execution of the instruction. Figures 6a and 6b illustrate a particular computation performed conventionally (Figure 6a) and using the computational method of the present invention (Figure 6b). In the illustrated example, the computational method of the invention results in a 20% reduction in computation time.

During a prior amendment, the claims had been amended to recite that one functional unit, during execution of an instruction, outputs data to or receives input data from another functional unit *in the midst of execution of the instruction*. This language was believed to be more express than the original language. Based on the Examiner's interpretation, however, the amended language introduced ambiguity into the claim. Accordingly, the claims have been amended to revert to the original language.

The O'Connor reference teaches something quite different. O'Connor relates to "scoreboarding," a method of resource arbitration in which a centralized set of tables ("registers") in the CPU keep track of what is being used and when. Each row indicates what is being used at each clock tick. Scoreboarding is used to manage dispatch, stalling, and completion of instructions, to watches for Write-After-Write, Read-After-Write and

Write-After-Read (WAW, RAW, WAR) hazards, and to manage the execution sequence (e.g. possible instruction reordering) to avoid these hazards. Hazards are detected by observing register references and operation types.


In particular, O'Connor describes a hardware-efficient implementation of a bypass circuit that enables a result from one execution unit to be provided to a waiting execution unit at the same time that result is sent to a register.

In O'Connor, one execution unit waits for results from another execution unit in order for the first execution unit to begin execution. The execution units do not operate concurrently to achieve execution of an instruction.

Accordingly, claims 1 and 4 are believed to patentably define over the cited references.

Dependent claims 2, 3, and 5 are also believed to add novel and patentable subject matter to their respective independent claims. Withdrawal of the rejection and allowance of claims 1-5 is respectfully requested.

Respectfully submitted,


Michael J. Ure, Reg. 33,089

Dated: September 1, 2005